

2.4 Requirements

2.4.1 Specification of the coded audio bitstream syntax

2.4.1.1 Audio sequence

Syntax	No. of bits	Mnemonic
audio sequence()		
{		
while (nextbits()==syncword) {		
frame()		
}		
}		

2.4.1.2 Audio frame

Syntax	No. of bits	Mnemonic
frame()		
{		
header()		
error_check()		
audio_data()		
ancillary_data()		
}		

2.4.1.3 Header

Syntax	No. of bits	Mnemonic
header()		
{		
syncword	**12**	**bslbf**
ID	**1**	**bslbf**
layer	**2**	**bslbf**
protection_bit	**1**	**bslbf**
bitrate_index	**4**	**bslbf**
sampling_frequency	**2**	**bslbf**
padding_bit	**1**	**bslbf**
private_bit	**1**	**bslbf**
mode	**2**	**bslbf**
mode_extension	**2**	**bslbf**
copyright	**1**	**bslbf**
original/copy	**1**	**bslbf**
emphasis	**2**	**bslbf**
}		

2.4.1.4 Error check

Syntax	No. of bits	Mnemonic
error_check()		
{		
if (protection_bit==0)		
crc_check	**16**	**rpchof**
}		

2.4.1.5 Audio data, Layer I

Syntax	No. of bits	Mnemonic
audio_data() {		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
allocation[ch][sb]	**4**	**uimsbf**
for (sb=bound; sb<32; sb++) {		
allocation[0][sb]	**4**	**uimsbf**
allocation[1][sb]=allocation[0][sb]		
}		
for (sb=0; sb<32; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
scalefactor[ch][sb]	**6**	**uimsbf**
for (s=0; s<12; s++) {		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
sample[ch][sb][s]	**2..15**	**uimsbf**
for (sb=bound; sb<32; sb++)		
if (allocation[0][sb]!=0)		
sample[0][sb][s]	**2..15**	**uimsbf**
}		
}		

2.4.1.6 Audio data, Layer II

Syntax	No. of bits	Mnemonic
audio_data()		
{		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
allocation[ch][sb]	**2..4**	**uimsbf**
for (sb=bound; sb<sblimit; sb++) {		
allocation[0][sb]	**2..4**	**uimsbf**
allocation[1][sb]=allocation[0][sb]		
}		
for (sb=0; sb<sblimit; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
scfsi[ch][sb]	**2**	**bslbf**
for (sb=0; sb<sblimit; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0) {		
if (scfsi[ch][sb]==0) {		
scalefactor[ch][sb][0]	**6**	**uimsbf**
scalefactor[ch][sb][1]	**6**	**uimsbf**
scalefactor[ch][sb][2]	**6**	**uimsbf**
}		
if ((scfsi[ch][sb]==1) || (scfsi[ch][sb]==3)) {		
scalefactor[ch][sb][0]	**6**	**uimsbf**
scalefactor[ch][sb][2]	**6**	**uimsbf**
}		
if (scfsi[ch][sb]==2)		
scalefactor[ch][sb][0]	**6**	**uimsbf**
}		
for (gr=0; gr<12; gr++) {		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0) {		
if (grouping[ch][sb])		
samplecode[ch][sb][gr]	**5..10**	**uimsbf**
else		
for (s=0; s<3; s++)		
sample[ch][sb][3*gr+s]	**3..16**	**uimsbf**
}		
for (sb=bound; sb<sblimit; sb++)		
if (allocation[0][sb]!=0) {		
if (grouping[0][sb])		
samplecode[0][sb][gr]	**5..10**	**uimsbf**
else		
for (s=0; s<3; s++)		
sample[0][sb][3*gr+s]	**3..16**	**uimsbf**
}		
}		
}		

2.4.1.7 Audio data, Layer III

Syntax	No. of bits	Mnemonic
audio_data()		
{		
main_data_begin	**9**	**uimsbf**
if (mode==single_channel)		
private_bits	**5**	**bslbf**
else		
private_bits	**3**	**bslbf**
for (ch=0; ch<nch; ch++)		
for (scfsi_band=0; scfsi_band<4; scfsi_band++)		
scfsi[ch][scfsi_band]	**1**	**bslbf**
for (gr=0; gr<2; gr++)		
for (ch=0; ch<nch; ch++) {		
part2_3_length[gr][ch]	**12**	**uimsbf**
big_values[gr][ch]	**9**	**uimsbf**
global_gain[gr][ch]	**8**	**uimsbf**
scalefac_compress[gr][ch]	**4**	**bslbf**
window_switching_flag[gr][ch]	**1**	**bslbf**
if (window_switching_flag[gr][ch]) {		
block_type[gr][ch]	**2**	**bslbf**
mixed_block_flag[gr][ch]	**1**	**uimsbf**
for (region=0; region<2; region++)		
table_select[gr][ch][region]	**5**	**bslbf**
for (window=0; window<3; window++)		
subblock_gain[gr][ch][window]	**3**	**uimsbf**
}		
else {		
for (region=0; region<3; region++)		
table_select[gr][ch][region]	**5**	**bslbf**
region0_count[gr][ch]	**4**	**bslbf**
region1_count[gr][ch]	**3**	**bslbf**
}		
preflag[gr][ch]	**1**	**bslbf**
scalefac_scale[gr][ch]	**1**	**bslbf**
count1table_select[gr][ch]	**1**	**bslbf**
}		
main_data ()		
}		

LUC 017196

The main data bitstream is defined below. The **main_data** field in the audio_data() syntax contains bytes from the main data bistream. However, because of the variable nature of Huffman coding used in Layer III, the main data for a frame does not generally follow the header and side information for that frame. The main_data for a frame starts at a location in the bitstream preceeding the header of the frame at a negative offset given by the value of main_data_begin. (See definition of main_data_begin and figure A.7.a).

Syntax	No. of bits	Mnemonic
main_data()		
{		
for (gr=0; gr<2; gr++) {		
for (ch=0; ch<nch; ch++) {		
if ((window_switching_flag[gr][ch]==1)		
&& (block_type[gr][ch]==2)) {		
if (mixed_block_flag[gr][ch]) {		
for (sfb=0; sfb<8; sfb++)		
scalefac_l[gr][ch][sfb]	**0..4**	**uimsbf**
for (sfb=3; sfb<12; sfb++)		
for (window=0; window<3; window++)		
scalefac_s[gr][ch][sfb][window]	**0..4**	**uimsbf**
}		
else {		
for (sfb=0; sfb<12; sfb++)		
for (window=0; window<3; window++)		
scalefac_s[gr][ch][sfb][window]	**0..4**	**uimsbf**
}		
}		
else {		
if ((scfsi[ch][0]==0) || (gr == 0))		
for(sfb=0;sfb<6;sfb++)		
scalefac_l[gr][ch][sfb]	**0..4**	**uimsbf**
if ((scfsi[ch][1]==0) || (gr == 0))		
for(sfb=6;sfb<11;sfb++)		
scalefac_l[gr][ch][sfb]	**0..4**	**uimsbf**
if ((scfsi[ch][2]==0) || (gr == 0))		
for(sfb=11;sfb<16;sfb++)		
scalefac_l[gr][ch][sfb]	**0..3**	**uimsbf**
if ((scfsi[ch][3]==0) || (gr == 0))		
for(sfb=16;sfb<21;sfb++)		
scalefac_l[gr][ch][sfb]	**0..3**	**uimsbf**
}		
Huffmancodebits()		
}		
}		
for (b=0; b<no_of_ancillary_bits; b++)		
ancillary_bit	**1**	**bslbf**
}		

Syntax	No. of bits	Mnemonic
Huffmancodebits() {		
for (i=0; i<big_values*2; i+=2) {		
hcod[ix][iy]	0..19	bslbf
if (ix==15 && linbits>0)		
linbitsx	1..13	uimsbf
if (x != 0)		
signx	1	bslbf
if (iy==15 && linbits>0)		
linbitsy	1..13	uimsbf
if (y != 0)		
signy	1	bslbf
is[i] = x		
is[i+1] = y		
}		
for (; i<big_values*2+count1*4; i+=4) {		
hcod[iv][iw][ix][iy]	1..6	bslbf
if (v!=0)		
signv	1	bslbf
if (w!=0)		
signw	1	bslbf
if (x!=0)		
signx	1	bslbf
if (y!=0)		
signy	1	bslbf
is[i] = v		
is[i+1] = w		
is[i+2] = x		
is[i+3] = y		
}		
for (; i<576; i++)		
is[i] = 0		
}		

2.4.1.8 Ancillary data

Syntax	No. of bits	Mnemonic
ancillary_data() {		
if ((layer == 1) || (layer == 2))		
for (b=0; b<no_of_ancillary_bits; b++)		
ancillary_bit	1	bslbf
}		

2.4.2 Semantics for the audio bitstream syntax

2.4.2.1 Audio sequence general

frame -- Layer I and Layer II: Part of the bitstream that is decodable by itself. In Layer I it contains information for 384 samples and in Layer II for 1 152 samples. It starts with a syncword, and ends just before the next syncword. It consists of an integer number of slots (four bytes in Layer I, one byte in Layer II).

-- Layer III: Part of the bitstream that is decodable with the use of previously acquired main information. In Layer III it contains information for 1 152 samples. Although the distance between the start of consecutive syncwords is an integer number of slots (one byte in Layer III), the audio information belonging to one frame is generally not contained between two successive syncwords.

2.4.2.2 Audio frame

header -- Part of the bitstream containing synchronization and state information.

error_check -- Part of the bitstream containing information for error detection.

audio_data -- Part of the bitstream containing information on the audio samples.

ancillary_data -- Part of the bitstream that may be used for ancillary data.

2.4.2.3 Header

The first 32 bits (four bytes) are header information which is common to all layers.

syncword -- The bit string '1111 1111 1111'.

ID -- One bit to indicate the ID of the algorithm. Equals '1' for ISO/IEC 11172-3 audio, '0' is reserved.

Layer -- 2 bits to indicate which layer is used, according to the following.

Layer	
'11'	Layer I
'10'	Layer II
'01'	Layer III
'00'	reserved

To change the layer, a reset of the audio decoder may be required.

protection_bit -- One bit to indicate whether redundancy has been added in the audio bitstream to facilitate error detection and concealment. Equals '1' if no redundancy has been added, '0' if redundancy has been added.

bitrate_index -- Indicates the bitrate. The all zero value indicates the 'free format' condition, in which a fixed bitrate which does not need to be in the list can be used. Fixed means that a frame contains either N or N+1 slots, depending on the value of the padding bit. The bitrate_index is an index to a table, which is different for the different layers.

The bitrate_index indicates the total bitrate irrespective of the mode (stereo, joint_stereo, dual_channel, single_channel).

bitrate_index	bitrate specified (kbits/s)		
	Layer I	Layer II	Layer III
'0000'	free	free	free
'0001'	32	32	32
'0010'	64	48	40
'0011'	96	56	48
'0100'	128	64	56
'0101'	160	80	64
'0110'	192	96	80
'0111'	224	112	96
'1000'	256	128	112
'1001'	288	160	128
'1010'	320	192	160
'1011'	352	224	192
'1100'	384	256	224
'1101'	416	320	256
'1110'	448	384	320
'1111'	forbidden	forbidden	forbidden

In order to provide the smallest possible delay and complexity, the decoder is not required to support a continuously variable bitrate when in Layer I or II. Layer III supports variable bitrate by switching the bitrate_index. The switching of the bitrate_index can be used either to optimize storage requirements on DSM or to interpolate any mean data rate by switching between nearby values in the bitrate table. However, in free format, fixed bitrate is required. The decoder is also not required to support bitrates higher than 448 kbits/s, 384 kbits/s, 320 kbits/s in respect to Layer I, II and III when in free format mode.

For Layer II, not all combinations of total bitrate and mode are allowed. See the following table.

bit rate (kbits/s)	Allowed modes
free format	all modes
32	single_channel
48	single_channel
56	single_channel.
64	all modes
80	single_channel
96	all modes
112	all modes
128	all modes
160	all modes
192	all modes
224	stereo, intensity stereo, dual channel
256	stereo, intensity stereo, dual channel
320	stereo, intensity stereo, dual channel
384	stereo, intensity stereo, dual channel

sampling_frequency -- Indicates the sampling frequency, according to the following table.

sampling_frequency	frequency specified (kHz)
'00'	44,1
'01'	48
'10'	32
'11'	reserved

A reset of the audio decoder may be required to change the sampling rate.

padding_bit -- If this bit equals '1', the frame contains an additional slot to adjust the mean bitrate to the sampling frequency, otherwise this bit will be '0'. Padding is necessary with a sampling frequency of 44,1 kHz. Padding may also be required in free format.

The padding should be applied to the bitstream such that the accumulated length of the coded frames, after a certain number of audio frames does not deviate more than (+0, -1 slot) from the following computed value:

$$\text{accumulated frame length} = \sum_{\text{first frame}}^{\text{current frame}} (\text{frame\_size} * \text{bitrate} / \text{sampling frequency})$$

where frame_size = 384 for Layer I,
1 152 for Layer II or III.

The following method can be used to determine whether or not to use padding:

```
for 1st audio frame:
        rest = 0;
        padding = no;

for each subsequent audio frame:
        if (Layer == 1) dif = (12 * bitrate) % sampling_frequency;
        else dif = (144 * bitrate) % sampling_frequency;
        rest = rest - dif;
        if (rest < 0) {
                padding = yes;
                rest = rest + sampling_frequency;
        }
        else padding = no;
```

private_bit -- Bit for private use. This bit will not be used in the future by ISO/IEC.

mode -- Indicates the mode according to the following table. In Layer I and II the joint_stereo mode is intensity_stereo, in Layer III it is intensity_stereo and/or ms_stereo.

mode	mode specified
'00'	stereo
'01'	joint_stereo (intensity_stereo and/or ms_stereo)
'10'	dual_channel
'11'	single_channel

In Layer I, in all modes except joint stereo, the value of bound equals 32. In layer II, in all modes except joint_stereo, the value of bound equals sblimit. In joint_stereo mode the bound is determined by the mode_extension.

mode_extension -- These bits are used in joint_stereo mode. In Layer I and II they indicate which subbands are in intensity_stereo. All other subbands are coded in stereo.

mode_extension	
'00'	subbands 4-31 in intensity_stereo, bound=4
'01'	subbands 8-31 in intensity_stereo, bound=8
'10'	subbands 12-31 in intensity_stereo, bound=12
'11'	subbands 16-31 in intensity_stereo, bound=16

In Layer III they indicate which type of joint stereo coding method is applied. The frequency ranges over which the intensity_stereo and ms_stereo modes are applied are implicit in the algorithm. For more information see 2.4.3.4.

mode_extension	intensity_stereo	ms_stereo
'00'	off	off
'01'	on	off
'10'	off	on
'11'	on	on

Note that the mode "stereo" is used if the mode bits specify stereo or equivalently if the mode bits specify joint stereo and the mode_extension specifies intensity_stereo "off" and ms_stereo "off".

copyright -- If this bit equals '0', there is no copyright on the ISO/IEC 11172-3 bitstream, '1' means copyright protected.

original/copy -- This bit equals '0' if the bitstream is a copy, '1' if it is an original.

emphasis -- Indicates the type of de-emphasis that shall be used.

emphasis	emphasis specified
'00'	none
'01'	50/15 microseconds
'10'	reserved
'11'	CCITT J.17

2.4.2.4 Error check

crc_check -- A 16 bit parity-check word is used for optional error detection within the encoded bitstream.

2.4.2.5 Audio data, Layer I

allocation[ch][sb] -- Indicates the number of bits used to code the samples in subband sb of channel ch. For subbands in intensity_stereo mode the bitstream contains only one allocation data element per subband.

allocation[ch][sb]	bits per sample
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	forbidden

Note: For code '0000' no samples are transferred.

scalefactor[ch][sb] -- Indicates the factor of subband sb of channel ch by which the requantized samples of subband sb in channel ch shall be multiplied. The six bits constitute an unsigned integer, index to table B.1 "Layer I, II scalefactors".

sample[ch][sb][s] -- Coded representation of the s-th sample in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the sample is valid for both channels.

2.4.2.6 Audio data, Layer II

allocation[ch][sb] -- Contains information concerning the quantizers used for the samples in subband sb in channel ch, whether the information on three consecutive samples has been grouped to one code, and on the number of bits used to code the samples. The meaning and length of this field depends on the number of the subband, the bitrate, and the sampling frequency. The bits in this field form an unsigned integer used as an index to the relevant table in table B.2 "Layer II bit allocation tables", which gives the number of levels used for quantization. For subbands in intensity_stereo mode the bitstream contains only one allocation data element per subband.

scfsi[ch][sb] -- Scalefactor selection information. This gives information on the number of scalefactors transferred for subband sb in channel ch and for which parts of the signal in this frame they are valid. The frame is divided into three equal parts of 12 subband samples each per subband.

scfsi[sb]	
'00'	three scalefactors transmitted, for parts 0,1,2 respectively.
'01'	two scalefactors transmitted, first one valid for parts 0 and 1, second one for part 2.
'10'	one scalefactor transmitted, valid for all three parts.
'11'	two scalefactors transmitted, first one valid for part 0, the second one for parts 1 and 2.

scalefactor[ch][sb][p] -- Indicates the factor by which the requantized samples of subband sb in channel ch and of part p of the frame should be multiplied. The six bits constitute an unsigned integer, index to table B.1 "Layer I, II scalefactors'.

grouping[ch][sb] -- Is a function that determines, whether grouping is in effect for coding of samples in subband sb of channel ch. Grouping means, that three consecutive samples (a triplet) of the current subband sb in channel ch in the current granule gr are coded and transmitted using one common codeword and not using three separate codewords. Grouping[ch][sb] is true, if in the Bit Allocation table currently in use (see B.2) the value found under the sb (row) and the allocation[sb] (column) is either 3, 5, or 9. Otherwise it is false. For subbands in intensity_stereo mode the grouping is valid for both channels.

samplecode[ch][sb][gr] -- Coded representation of the three consecutive samples in the granule gr in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the samplecode is valid for both channels.

sample[ch][sb][s] -- Coded representation of the s-th sample in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the sample is valid for both channels.

2.4.2.7 Audio data, Layer III

main_data_begin -- The value of main_data_begin is used to determine the location of the first bit of main data of a frame. The main_data_begin value specifies the location as a negative offset in bytes from the first byte of the audio sync word. The number of bytes belonging to the header and side information is not taken into account. For example, if main_data_begin == 0, then main data starts after the side information. Examples are given in figure A.7.a and figure A.7.b.

private_bits -- Bits for private use. These bits will not be used in the future by ISO/IEC. The number of private_bits depends on the number of channels. The number of bits allocated for private_bits is determined to equalize the total number of bits used for side-information.

scfsi[ch][scfsi_band] -- In Layer III, the scalefactor selection information works similarly to audio Layer II. The main difference is the use of the variable scfsi_band to apply scfsi to groups of scalefactors instead of single scalefactors. The application of scalefactors to granules is controlled by scfsi.

scfsi[scfsi_band]	
'0'	scalefactors are transmitted for each granule
'1'	scalefactors transmitted for granule 0 are also valid for granule 1

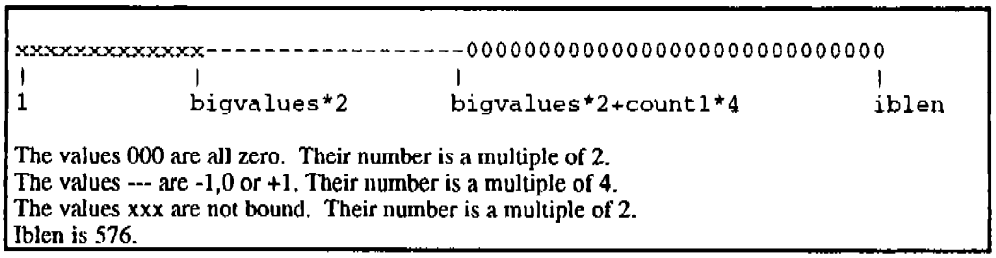
If short windows are switched on, i.e. block_type==2 for one of the granules, then scfsi is always 0 for this frame.

scfsi_band controls the use of the scalefactor selection information for groups of scalefactors (scfsi_bands).

scfsi_band	scalefactor bands (see table B.8)
0	0,1,2,3,4,5,
1	6,7,8,9,10,
2	11 ... 15
3	16 ... 20

part2_3_length[gr][ch] -- This value contains the number of main_data bits used for scalefactors and Huffman code data. Because the length of the side information is always the same, this value can be used to calculate the beginning of the main information for the next granule or the position of the ancillary information (if used). Note that single channel audio frames contain 17 bytes of side information and dual channel audio frames contain 32 bytes of side information (see 2.4.1.7 Audio Data, Layer III - syntax for audio_data()).

big_values[gr][ch] -- The spectral values of each granule are coded with different Huffman code tables. The full frequency range from zero to the Nyquist frequency is divided into several regions, which then are coded using different tables. Partitioning is done according to the maximum quantized values. This is done with the assumption that values at higher frequencies are expected to have lower amplitudes or do not need to be coded at all. Starting at high frequencies, the pairs of quantized values equal to zero are counted. This number is named "rzero". Then, quadruples of quantized values with absolute value not exceeding 1 (i.e. only 3 possible quantization levels) are counted. This number is named "count1". Again an even number of values remain. Finally, the number of pairs of values in the region of the spectrum which extends down to zero is named "big_values". The maximum absolute value in this range is constrained to 8191. The following figure shows the partitioning:



The values 000 are all zero. Their number is a multiple of 2.
The values --- are -1,0 or +1. Their number is a multiple of 4.
The values xxx are not bound. Their number is a multiple of 2.
Iblen is 576.

global_gain[gr][ch] -- The quantizer step size information is transmitted in the side information variable global_gain. It is logarithmically quantized. For the application of global_gain, refer to the formula in 2.4.3.4, "Formula for requantization and all scaling".

scalefac_compress[gr][ch] -- Selects the number of bits used for the transmission of the scalefactors according to the following table:

if **block_type** is 0, 1, or 3:
slen1: length of scalefactors for the scalefactor bands 0 to 10
slen2: length of scalefactors for the scalefactor bands 11 to 20

if **block_type** is 2 and mixed_block_flag is 0:
slen1: length of scalefactors for the scalefactor bands 0 to 5
slen2: length of scalefactors for the scalefactor bands 6 to 11

LUC 017204

if **block_type** is 2 and mixed_block_flag is 1:
slen1: length of scalefactors for the scalefactor bands 0 to 7 (long window scalefactor band) and 3 to 5 (short window scalefactor band) Note: Scalefactor bands 0-7 are from the "long window

scalefactor band" table, and scalefactor bands 3 to 11 from the "short window scalefactor band" table. This combination of partitions is contiguous and spans the entire frequency spectrum.
slen2: length of scalefactors for the scalefactor bands 6 to 11

scalefac_compress[gr]	slen1	slen2
0	0	0
1	0	1
2	0	2
3	0	3
4	3	0
5	1	1
6	1	2
7	1	3
8	2	1
9	2	2
10	2	3
11	3	1
12	3	2
13	3	3
14	4	2
15	4	3

window_switching_flag[gr][ch] -- Signals that the block uses an other than normal (type 0) window.

If window_switching_flag is set, several other variables are set by default:

region0_count = 7 (in case of **block_type==1** or **block_type==3** or **block_type==2** and mixed_block_flag)
region0_count = 8 (in case of **block_type==2** and not mixed_block_flag)
region1_count = 36 Thus all remaining values in the big_value region are contained in region 1.

If window_switching_flag is not set, then the value of **block_type** is zero.

block_type[gr][ch] -- Indicates the window type for the granule (see description of the filterbank, Layer III).

block_type[gr]	
0	reserved
1	start block
2	3 short windows
3	end block

Block_type and **mixed_block_flag** give the information about assembling of values in the block and about length and count of the transforms (see figure A.4 for a schematic, annex C for an analytic description). If window_switching_flag==1, then the mixed_block_flag indicates whether lower frequency polyphase filter subbands are coded using normal window type. The polyphase filterbank is described in 2.4.3.

In the case of long blocks (block_type not equal to 2 or in the lower subbands of block_type 2 if the mixed_block_flag is set) the IMDCT generates an output of 36 values every 18 input values. The output is windowed depending on the block_type and the first half is overlapped with the second half of the block before. The resulting vector is the input of the synthesis part of the polyphase filterbank of one band.

In the case of short blocks (in the upper subbands of a type 2 block if the mixed_block_flag is set, or in all subbands of a type 2 block if mixed_block_flag is not set), three transforms are performed producing 12 output values each. The three vectors are windowed and overlapped each. Concatenating 6 zeros on both ends of the resulting vector gives a vector of length 36, which is processed like the output of a long transform.

mixed_block_flag[gr][ch] -- Indicates that lower frequencies are transformed with a window type that is different than that which is used at higher frequencies. If mixed_block_flag is zero, then all blocks are transformed as indicated by block_type[gr][ch]. If mixed_block_flag is one, then the frequency lines corresponding to the two lowest frequency polyphase subbands are transformed with normal window (block_type==0), while the remaining 30 subbands are transformed as block_type[gr][ch].

table_select[gr][ch][region] -- Different Huffman code tables are used depending on the maximum quantized value and the local statistics of the signal. There are a total of 32 possible tables given in table B.7.

subblock_gain[gr][ch][window] -- Indicates the gain offset (quantization: factor 4) from the global gain for one subblock. Used only with block type 2 (short windows). The values of the subblock have to be divided by $4^{(\text{subblock\_gain[window]})}$ in the decoder. See 2.4.3.4 - Formula for requantization and all scaling.

region0_count[gr][ch] -- A further partitioning of the spectrum is used to enhance the performance of the Huffman coder. It is a subdivision of the region which is described by big_values. The purpose of this subdivision is to get better error robustness and better coding efficiency. Three regions are used, they are named: region 0, 1 and 2. Each region is coded using a different Huffman code table depending on the maximum quantized value and the local signal statistics.

The values region0_count and region1_count are used to indicate the boundaries of the regions. The region boundaries are aligned with the partitioning of the spectrum into scale factor bands.

The field region0_count contains one less than the number of scalefactor bands in region 0. In the case of short blocks, each scale factor band is counted three times, once for each short window, so that a region0_count value of 8 indicates that region1 begins at scalefactor band number 3.

If block_type==2 and mixed_block_flag==0, the total amount of scalefactor bands for the granule in this case is 12*3=36. If block_type==2 and mixed_block_flag==1, the amount of scalefactor bands is 8+9*3=35. If block_type!=2, the amount of scalefactor bands is 21.

region1_count[gr][ch] -- region1_count counts one less than the number of scalefactor bands in region 1. Again, if block_type==2 the scalefactor bands representing different time slots are counted separately.

preflag[gr][ch] -- This is a shortcut for additional high frequency amplification of the quantized values. If preflag is set, the values of a table are added to the scalefactors (see table B.6). This is equivalent to multiplication of the requantized scalefactors with table values. If block_type==2 (short blocks) preflag is never used.

scalefac_scale[gr][ch] -- The scalefactors are logarithmically quantized with a step size of 2 or ($\sqrt{2}$) depending on scalefac_scale. The following table indicates the scale factor multiplier used in the requantization equation for each stepsize.

scalefac_scale[gr]	scalefac_multiplier
0	0,5
1	1

count1table_select[gr][ch] -- This flag selects one of two possible Huffman code tables for the region of quadruples of quantized values with magnitude not exceeding 1.

count1table_select[gr]	
0	Table B.7 - A
1	Table B.7 - B

scalefac_l[gr][ch][sfb], scalefac_s[gr][ch][sfb][window], is_pos[sfb] -- The scalefactors are used to colour the quantization noise. If the quantization noise is coloured with the right shape, it is masked completely. Unlike Layers I and II, the Layer III scalefactors say nothing about the local maximum of the quantized signal. In Layer III, scalefactors are used in the decoder to get division factors for groups of values. In the case of Layer III, the groups stretch over several frequency lines. These groups are called scalefactor bands and are selected to resemble critical bands as closely as possible.

The scalefac_compress table shows that the scalefactors 0...10 have a range of 0 to 15 (maximum length 4 bits) and the scalefactors 11...21 have a range of 0 to 7 (maximum length 3 bits).

If intensity_stereo is enabled (modebit_extension) the scalefactors of the "zero_part" of the difference (right) channel are used as intensity_stereo positions, is_pos[sfb] (see 2.4.3.4, MS_stereo mode). is_pos[sfb] is the intensity stereo position for scalefactor band sfb.

The subdivision of the spectrum into scalefactor bands is fixed for every block length and sampling frequency and stored in tables in the coder and decoder (see table B.8). The scale factor for frequency lines above the highest line in the tables is zero, which means that the actual multiplication factor is 1,0.

The scalefactors are logarithmically quantized. The quantization step is set with scalefac_scale.

huffmancodebits() -- Huffman encoded data.

The syntax for huffmancodebits() shows how quantized values are encoded. Within the big_values partition, pairs of quantized values with an absolute value less than 15 are directly coded using a Huffman code. The codes are selected from huffman tables 0 through 31 in table B.7. Always pairs of values (x,y) are coded. If quantized values of magnitude greater than or equal to 15 are coded, the values are coded with a separate field following the huffman code. If one or both values of a pair is not zero, one or two sign bits are appended to the code word.

The Huffman tables for the big_values partition are comprised of three parameters:

hcod[lxl][lyl]	is the Huffman code table entry for values x,y.
hlen[lxl][lyl]	is the Huffman length table entry for values x,y.
linbits	is the length of linbitsx or linbitsy when they are coded.

The syntax for huffmancodebits contains the following fields and parameters:

signv	is the sign of v (0 if positive, 1 if negative).
signw	is the sign of w (0 if positive, 1 if negative).
signx	is the sign of x (0 if positive, 1 if negative).
signy	is the sign of y (0 if positive, 1 if negative).
linbitsx	is used to encode the value of x if the magnitude of x is greater or equal to 15. This field is coded only if lxl in hcod is equal to 15. If linbits is zero, so that no bits are actually coded when lxl==15, then the value linbitsx is defined to be zero.
linbitsy	is the same as linbitsx but for y.
is[l]	Is the quantized value for frequency line number l.

The linbitsx or linbitsy fields are only used if a value greater or equal to 15 needs to be encoded. These fields are interpreted as unsigned integers and added to 15 to obtain the encoded value. The linbitsx and linbitsy fields are never used if the selected table is one for blocks with a maximum quantized value less than 15. Note that a value of 15 can still be encoded with a huffman table for which linbits is zero. In this case, the linbitsx or linbitsy fields are not actually coded, since linbits is zero.

Within the count1 partition, quadruples of values with magnatude less than or equal to one are coded. Again magnitude values are coded using a Huffman code from tables A or B in table B.7. Again, for each non-zero value, a sign bit is appended after the huffman code symbol.

The Huffman tables for the count1 partition are comprised of the following parameters:

hcod[lvl][lwl][lxl][lyl]	is the Huffman code table entry for values v,w,x,y.
hlen[lvl][lwl][lxl][lyl]	is the Huffman length table entry for values v,w,x,y.

Huffman code table B is not really a 4-dimensional code because it is constructed from the trivial code: 0 is coded with a 1, and 1 is coded with a 0.

Quantized values above the count1 partition are zero, so they are not encoded.

For clarity, the parameter "count1" is used in this document to indicate the number of Huffman codes in the count1 region. However, unlike the bigvalues partition, the number of values in the count1 partition is not

explicitly coded by a field in the syntax. The end of the count1 partition is known only when all bits for the granule (as specified by part2_3_length), have been exhausted, and the value of count1 is known implicitly after decoding the count1 region.

The order of the Huffman data depends on the block_type of the granule. If block_type is 0, 1 or 3 the Huffman encoded data is ordered in terms of increasing frequency.

If block_type==2 (short blocks) the Huffman encoded data is ordered in the same order as the scalefactor values for that granule. The Huffman encoded data is given for successive scalefactor bands, beginning with scalefactor band 0. Within each scalefactor band, the data is given for successive time windows, beginning with window 0 and ending with window 2. Within each window, the quantized values are then arranged in order of increasing frequency.

2.4.2.8 Ancillary data

Ancillary_bit -- User definable.

The number of ancillary bits (no_of_ancillary_bits) equals the available number of bits in an audio frame minus the number of bits actually used for header, error check and audio data. In Layer I and II the no_of_ancillary_bits corresponds to the distance between the end of the audio data and the beginning of the next header. In Layer III the no_of_ancillary_bits corresponds to the distance between the end of the Huffman_code_bits and the location in the bitstream where the next frame's main_data_begin pointer points to.

2.4.3 The audio decoding process

2.4.3.1 General

The first action is synchronization of the decoder to the incoming bitstream. Just after startup this may be done by searching in the bitstream for the 12 bit syncword. In some applications the ID, layer, and protection status are already known to the decoder, and thus the first 16 bits of the header should be regarded as a 16 bit syncword, thereby allowing a more reliable synchronization. The position of consecutive syncwords can be calculated from the information provided by the seven bits after the protection_bit: the bitstream is subdivided in slots. The distance between the start of two consecutive syncwords is equal to "N" or "N+1" slots. The value of "N" depends on the layer.

For Layer I the following equation is valid:

$$N = 12 * \frac{\text{bitrate}}{\text{sampling\_frequency}}.$$

For Layers II and III the equation becomes:

$$N = 144 * \frac{\text{bitrate}}{\text{sampling\_frequency}}.$$

If this calculation does not give an integer number the result is truncated and 'padding' is required. In this case the number of slots in a frame will vary between N and N+1. The padding bit is set to '0' if the number of slots equals N, and to '1' otherwise. This knowledge of the position of consecutive syncwords greatly facilitates synchronization.

If the bitrate index equals '0000', the exact bitrate is not indicated. N can be determined from the distance between consecutive syncwords and the value of the padding bit.

The mode bits in the bitstream shall be read and if their value is '01' the mode_extension bits shall also be read. The mode_extension bits set the 'bound' as shown in 2.4.2.3 and thus indicate which subbands are coded in joint_stereo mode.

If the protection bit in the header equals '0', a CRC-check word has been inserted in the bitstream just after the header. The error detection method used is 'CRC-16' whose generator polynomial is:

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

The bits included into the CRC-check are given by table B.5.

The method is depicted in figure A.9 "CRC-check diagram". The initial state of the shift register is '1111 1111 1111 1111'. Then all the bits included into the CRC-check are input to the circuit shown in figure A.9 "CRC-check diagram". After each bit is input the shift register is shifted by one bit. After the last shift operation, the outputs $b_{15}...b_0$ constitute a word to be compared with the CRC-check word in the bitstream. If the words are not identical, a transmission error has occurred in the protected field of the bitstream. To avoid annoying distortions, application of a concealment technique, such as muting of the actual frame or repetition of the previous frame, is recommended.

2.4.3.2 Layer I

After the part of the decoding which is common to all layers (see 2.4.3.1) the bit allocation information has to be read for all subbands, and the scalefactors read for all subbands with a nonzero bit allocation. The decoder flowchart is given in figure A.1 "Layer I and II decoder flow chart".

2.4.3.2.1 Requantization of subband samples

From the bit allocation the number of bits nb that has to be read for the samples in each subband is known. The order of the samples is given in 2.4.1.5 for each mode. After the bits for one sample have been gathered from the bitstream, the first bit has to be inverted. The resulting number can be considered as a two's

complement fractional number, where the MSB represents the value -1. The requantized value can be obtained by applying a linear formula :

$$s'' = \frac{2^{nb}}{2^{nb} - 1} * \left(s''' + 2^{-nb+1}\right)$$

where

s''' is the fractional number;
s'' is the requantized value;
nb is the number of bits allocated to samples in the subband.

Samples in subbands which are in intensity_stereo mode must be copied to both channels. The requantized value has to be rescaled. The multiplication factor can be found in the table B.1 "Layer I, II scalefactors". The rescaled value s' is calculated as :

$$s' = \text{factor} * s''$$

2.4.3.2.2 Synthesis subband filter

If a subband has no bits allocated to it, the samples in that subband are set to zero. Each time the subband samples for all 32 subbands of one channel have been calculated, they can be applied to the synthesis subband filter and 32 consecutive audio samples can be calculated. The actions in flow diagram figure A.2 "Synthesis subband filter flow chart" show the reconstruction operation. The coefficients N_{ik} for the matrixing operation are given by

$$N_{ik} = \cos\left[(16 + i)(2k+1)\frac{\pi}{64}\right] \qquad 0 \le i \le 63,\ 0 \le k \le 31$$

The coefficients D_i for the windowing operation can be found in table B.3 "Coefficients D_i of the synthesis window". The coefficients have been derived by numerical optimization. One frame contains 12 * 32 = 384 subband samples, which result, after filtering, in 384 audio samples.

2.4.3.3 Layer II

Layer II is a more efficient but more complex coding scheme than Layer I. The flowchart in figure A.1 "Layer I and II decoder flow chart" applies to both Layers I and II. The first step is to perform the decoding which is common to all three layers (see 2.4.3.1).

2.4.3.3.1 Bit allocation decoding

For different combinations of bitrate and sampling frequency different bit allocation tables exist (table B.2 "Layer II bit allocation tables"). Note that the bitrates given in the table headers are per channel. If the mode is not single_channel, the bitrate should be divided by two to obtain the bitrate per channel. The decoding of the bit allocation table is done in a three-step approach. The first step consists of reading 'nbal' (2,3, or 4) bits of information for one subband from the bitstream. The value of 'nbal' is given in the second column of the relevant table B.2 "Layer II bit allocation tables". These bits shall be interpreted as an unsigned integer number. The second step uses this number and the number of the subband as indices to point to a value in the table. This value represents the number of levels 'nlevels' used to quantize the samples in the subband. As a third step, using table B.4 "Layer II classes of quantization", the number of bits used to code the quantized samples, the requantization coefficients, and whether the codes for three consecutive subband samples have been grouped to one code can be determined. It can be seen from the bit allocation tables that some of the highest subbands will never have bits allocated. The number of the lowest subband that will not have bits allocated to it is assigned to the identifier 'sblimit'.

2.4.3.3.2 Scalefactor selection information decoding

The 36 samples in one subband within a frame are divided in three equal parts of 12 subband samples. Each part can have its own scalefactor. The number of scalefactors that has to be read from the bitstream depends on scfsi[sb]. The scalefactor selection information scfsi[sb] is read from the bitstream for the subbands that have a nonzero bit allocation. If scfsi[sb] equals '00' three scalefactors are transmitted, for parts 0,1,2 respectively. If scfsi[sb] equals '01' two scalefactors are transmitted, the first one valid for parts 0 and 1, the

second one for part 2. If scfsi[sb] equals '10' one scalefactor is transmitted, valid for all three parts. If scfsi[sb] equals '11' two scalefactors are transmitted, the first one valid for part 0, the second one for parts 1 and 2.

2.4.3.3.3 Scalefactor decoding

For every subband with a nonzero bit allocation the coded scalefactors for that subband are read from the bitstream. The number of coded scalefactors and the part of the subband samples they refer to is defined by scfsi[sb]. The 6 bits of a coded scalefactor should be interpreted as an unsigned integer index to table B.1 "Layer I, II scalefactors". This table gives the scalefactor by which the relevant subband samples should be multiplied after requantization.

2.4.3.3.4 Requantization of subband samples

Next the coded samples are read. As can be seen from 2.4.1.6, the coded samples appear as triplets, the code contains three consecutive samples at a time. From table B.4 "Layer II classes of quantization" it is known how many bits have to be read for one triplet from the bitstream for each subband. Also from table B.4 "Layer II classes of quantization", it is known whether this code consists of three consecutive separable codes for each sample or of one combined code for the three samples (grouping). In the last case degrouping must be performed. The combined code has to be regarded as an unsigned integer, called 'c'. The following algorithm will supply the three separate codes s[0], s[1], s[2].

```
for (i=0; i<3; i++) {
        s[i]= c % nlevels
        c   = c DIV nlevels
}
```

where nlevels is the number of steps as shown in table B.2 "Layer II bit allocation table".

The first bit of each of the three codes has to be inverted, and the resulting numbers should be regarded as two's complement fractional numbers, where the MSB represents the value -1. The requantized values can be obtained by applying a linear formula :

$$s'' = C * (s''' + D)$$

where

s''' is the fractional number;

s'' is the requantized value.

The values of the constants C and D are given in table B.4 "Layer II classes of quantization". The requantized values have to be rescaled. The multiplication factors can be found in the table B.1 "Layer I, II scalefactors". as described above. The rescaled value s' is calculated as :

$$s' = \text{factor} * s''$$

2.4.3.3.5 Synthesis subband filter

If a subband has no bits allocated to it, the samples in that subband are set to zero. Each time the subband samples for all 32 subbands of one channel have been calculated, they can be applied to the synthesis subband filter and 32 consecutive audio samples can be calculated. For that purpose, the actions in the flow diagram of figure A.2 "Synthesis subband filter flow chart" have to be carried out. The coefficients N_{ik} for the matrixing operation are given by

$$N_{ik} = \cos\left[(16 + i)(2k+1)\frac{\pi}{64}\right] \qquad 0 \le i \le 63,\ 0 \le k \le 31$$

The coefficients D_i for the windowing operation can be found in table B.3 "Coefficients D_i of the synthesis window". One frame contains 36 * 32 = 1 152 subband samples, which result after filtering in 1 152 audio samples.

2.4.3.4 Layer III

Additional frequency resolution is provided by the use of an hybrid filterbank. Every band is split into 18 frequency lines by use of an MDCT. The window length of the MDCT is 36. Adaptive window switching is used to control time artifacts (pre-echoes), see the description in annex C. The frequency above which shorter blocks (better time resolution) are used can be selected. Parts of the signal below a frequency depending on "mixed_block_flag" are coded with better frequency resolution, parts of the signal above are coded with better time resolution.

The frequency components are quantized using a nonuniform quantizer and coded using a Huffman encoder. The Huffman coder uses one of 18 different tables (see table B.7). A buffer is used to help enhance the coding efficiency of the Huffman coder and to help in the case of pre-echo conditions (see the description in annex C). The size of the input buffer is the size of one frame at the bitrate of 160 kbits/s per channel for Layer III. The short term buffer technique used is called "bit reservoir" because it uses short-term variable bitrate with a maximum integral offset from the mean bitrate.

Each frame holds the data from 2 granules. The audio data in a frame is allocated in the following way:

- main_data_begin pointer
- side info for both granules (scfsi)
- side info granule 1
- side info granule 2

The header and this part of the audio data constitute the side information stream.

- scalefactors and Huffman code data granule 1
- scalefactors and Huffman code data granule 2
- ancillary data

These data constitute the main data stream. The main_data_begin pointer specifies a negative offset from the position of the first byte of the header.

2.4.3.4.1 Decoding

The first action is the synchronization of the decoder to the incoming bitstream. This is done as in the other layers. The header information (first 32 bits including syncword) is read in just as in the other layers. The information about sampling frequency is used to select the scalefactor_band table (see annex B.8).

2.4.3.4.2 Side information

The side information must be extracted from the bitstream and stored for use during the decoding of the associated frame. The table select information is used to select the Huffman decoder table and the number of ESC-bits (linbits), according to table B.7.

2.4.3.4.3 Start of main_data

The main_data (scalefactors, Huffman coded data and ancillary information) are not necessarily located adjacent to the side information. This is described in figure A.7.a and figure A.7.b. The beginning of the main data part is located by using the main_data_begin pointer of the current frame. The allocation of the main data is done in a way that all main data are resident in the input buffer when the header of the next frame is arriving in the input buffer. The decoder has to skip Header and side information when decoding the main data. It knows their positions from the bitrate_index and padding_bit. The length of the Header is always 4 bytes, the length of the side information is 17 bytes in mode single_channel and 32 bytes in the other modes. Main data can span more than one block of header and side information (see figure A.7.b).

2.4.3.4.4 Buffer considerations

The following rule can be used to calculate the maximum number of bits used for one granule:

The buffer length is 7 680 bits. This value is used as the maximum buffer at every bitrate. At the highest possible bitrate of Layer III (320 kbits/s per stereo signal) and sampling frequency 48 kHz the mean frame length is (320 000/48 000)*1 152 = 7 680 bits. Therefore the frames must be of constant length at this bitrate and sampling frequency. At 64 kbits/s (128 kbits/s stereo) the mean granule length is